Modul Klasifikasi Aduan dengan Pendekatan Kemiripan Teks pada Aplikasi Perangkat Bergerak Suara Warga (Surga) Kota Kediri

Tegar Rachman Muzzammil, R. V. Hari Ginardi, dan Diana Purwitasari
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia
e-mail: hari@its-sby.edu

***Abstrak*—Sistem informasi layanan masyarakat merupakan sistem informasi yang akhir-akhir ini mulai diterapkan di berbagai daerah. Salah satunya adalah Kota Kediri yang memiliki sistem layanan pengaduan masyarakat yang bernama Suara Warga (SURGA) Kota Kediri. Dalam penerimaan aduan, terkadang aduan yang masuk ke dalam sistem memiliki kemiripan dengan aduan yang sudah ada. Hal ini dikarenakan adanya kemungkinan pengadu mengirimkan aduan berulang kali atau beberapa pengadu mengirimkan aduan dengan isi yang sama. *Manhattan similarity* adalah salah satu algoritma yang digunakan untuk mendeteksi kemiripan dua dokumen. *Manhattan similarity* dapat diimplementasikan pada aduan yang masuk ke dalam sistem. Aduan yang masuk diproses dengan pendekatan *text similarity*, yaitu *text processing* dan dimodelkan dalam bentuk *vector space model* sehingga dapat dihitung jarak antar aduan menggunakan *Manhattan distance*. Perhitungan jarak antar aduan dibatasi dengan penyusunan *cluster* menggunakan *K-Means clustering*, sehingga hanya aduan yang berada pada *cluster* atau klasifikasi yang sama yang dibandingkan. Uji coba dilakukan dengan menyusun *cluster* yang bertujuan untuk klasifikasi aduan dan dilakukan deteksi kemiripan. Setelah klasifikasi dan deteksi dilakukan, sejumlah aduan diambil dari setiap *cluster* dan ditanyakan kepada 15 responden. Hasil uji coba menunjukkan bahwa aduan dapat dideteksi kemiripannya dengan jarak *Manhattan distance* minimal 0,9993 antar aduan dengan tingkat akurasi untuk aduan tidak mirip 100% dan untuk aduan mirip 90%. Waktu total yang dibutuhkan untuk melakukan proses klasifikasi dan deteksi kemiripan teks adalah 17 menit 27 detik dengan jumlah aduan 387.**

***Kata Kunci*—K-Means Clustering, Manhattan Distance, Manhattan Similarity, Text Processing, TF-IDF, Vector Space Model**

I. PENDAHULUAN

KOTA Kediri adalah sebuah kota yang terletak di Provinsi Jawa Timur, Indonesia. Kota Kediri merupakan kota yang memiliki potensi untuk berkembang. Berdasarkan Dinas Sosial dan Tenaga Kerja Kota Kediri jumlah penduduk di Kota Kediri pada tahun 2013 mencapai 267.310 orang [1].

Dengan meningkatnya jumlah penduduk di Kota Kediri, tentulah diperlukan wadah yang dapat menampung aduan serta aspirasi warga. Kota Kediri sudah memiliki wadah yang bernama SURGA (Suara Warga) Kota Kediri. SURGA Kota Kediri ini dapat menjadi wadah untuk menampung aduan serta aspirasi warga melalui SMS atau dari situsnya secara langsung.

Isi dari aduan yang disampaikan melalui SURGA ini tidak mempunyai format aduan yang terikat. Selain itu, dengan banyaknya aduan setiap hari, sangatlah sulit untuk mengklasifikasi jenis aduan dari warga apabila dilakukan secara manual. Hal ini dikarenakan manusia perlu membaca isi dari tiap aduan dan kemudian memberikan kategori dari aduan tersebut. Selain itu, aduan juga dapat masuk pada malam hari di saat manusia istirahat, hal ini menunda klasifikasi aduan yang masuk.

Klasifikasi dokumen teks adalah permasalahan yang mendasar dan penting. Di dalam dokumen teks, tulisan yang terkandung adalah bahasa alami manusia, yang merupakan bahasa dengan struktur yang kompleks dan jumlah kata yang sangat banyak [2].

Pada tugas akhir ini akan diangkat sebuah topik mengenai klasifikasi aduan dan deteksi kemiripan yang masuk ke dalam sistem SURGA Kota Kediri. Topik ini dipilih untuk mendeteksi duplikasi serta aduan yang mirip dengan aduan yang sudah masuk, dan juga mengklasifikasikan kategori pada setiap aduan yang masuk. Dengan menggunakan mesin, aduan yang masuk akan diproses secara otomatis.

Dengan modul aplikasi ini nantinya modul ini dapat mendeteksi kemiripan aduan yang masuk, sehingga apabila terdapat aduan yang mirip dengan aduan yang masuk sebelumnya, maka akan dianggap aduan yang sama, sehingga tidak perlu ditampilkan ulang aduan-aduan yang bersifat mirip.

Setelah itu, modul ini nantinya akan mengklasifikasikan aduan yang masuk. Aduan dapat ditampilkan sesuai dengan klasifikasi yang telah dilakukan.

II. URAIAN PENELITIAN

A. *Text Processing*

Text mining merupakan proses pengambilan data berupa teks dari sebuah sumber dalam hal ini sumbernya adalah dokumen. Dengan *text mining* dapat dicari kata-kata kunci yang dapat mewakili isi dari suatu dokumen lalu dianalisa dan dilakukan pencocokan antara dokumen dengan basis data kata kunci yang telah dibuat. *Text processing* merupakan bagian

dari *text mining* [3]. Tahapan *text processing* secara umum adalah *tokenizing*, *stopping*, dan *stemming*.

1) Tokenizing

Tokenizing merupakan proses yang dilakukan pada suatu dokumen untuk mendapatkan *term-term*. Proses yang dilakukan adalah memotong kata-kata yang membangun suatu dokumen dan hasil dari potongan disebut *token*, dan mungkin dalam proses yang sama membuang berbagai karakter seperti tanda baca [4].

2) Stopping

Stopping merupakan proses yang dilakukan setelah *tokenizing* pada *text processing.* Proses dari *stopping* adalah menghilangkan kata yang sering muncul pada umumnya yang disebut *stopword*. *Stopword* cenderung memiliki bobot yang rendah, sehingga hampir tidak mempengaruhi perhitungan apabila *stopword* dihapus. Salah satu teknik yang biasa digunakan untuk mengurangi indeks kata adalah dengan *stemming* atau menghilangkan *stopword* [5].

3) Stemming

Stemming merupakan proses untuk mendapatkan kata dasar dari suatu *term*. Tujuan dari proses ini dilakukan agar makna suatu *term* dari satu dokumen sama dengan dokumen lainnya karena *term* tersebut sudah berada pada bentuk dasar. Karena alasan adanya transformasi kata, suatu dokumen biasanya menggunakan kata yang berbeda bentuk, padahal kata tersebut memiliki makna yang tidak jauh berbeda. Dalam banyak situasi, akan sangat membantu apabila kata yang berbeda bentuknya tersebut dianggap sama [4]. Algoritma *stemmer* yang digunakan berdasarkan algoritma *Porter Stemmer*. *Porter Stemmer* dipilih dengan pertimbangan bahwa inti dari *stemmer* tersebut cocok dengan struktur kata pada Bahasa Indonesia [6]. Cara kerja algoritma ini adalah sebagai berikut:

- Apabila kata memiliki lebih dari dua suku kata, maka periksa akhiran partikel kata tersebut. Apabila kata tersebut memiliki akhiran partikel seperti *-kah*, *-lah*, atau *–pun*, maka hapus kata akhiran partikel dan kurangi satu suku kata.
- Apabila kata tersebut masih memiliki jumlah suku lebih dari dua, maka periksa akhiran milik kata tersebut. Apabila kata tersebut memiliki akhiran milik seperti *-ku* atau *–nya*, maka hapus kata akhiran milik tersebut dan kurangi satu suku kata.
- Apabila kata tersebut masih memiliki jumlah suku kata lebih dari dua, maka periksa kata awalan pertama (*first order prefix*) kata tersebut. Apabila kata tersebut memiliki kata awalan pertama seperti *meng-*, *meny-*, *men-*, *mem-*, *me-*, *peng-*, *peny-*, *pen-*, *pem-*, *di-*, *ter-* atau *ke-*, maka hapus kata awalan tersebut dan kurangi satu suku kata.
- Apabila kata tersebut masih memiliki jumlah suku kata lebih dari dua, maka periksa kata awalan kedua (*second order prefix*) kata tersebut. Apabila kata tersebut memiliki kata awalan kedua seperti *ber-*, *be-*, *per-* atau *pe-*, maka hapus kata awalan tersebut dan kurangi satu suku kata.
- Apabila kata tersebut masih memiliki jumlah suku kata lebih dari dua, maka periksa kata akhiran (*suffix*) kata tersebut. Apabila kata tersebut memiliki kata akhiran seperti *–kan*, *-an* atau *-i*, maka hapus kata akhiran tersebut dan kurangi satu suku kata.
- Kata dasar ditemukan.

B. Term Frequency-Inversed Document Frequency Algorithm (TF-IDF)

Salah satu cara untuk memberi bobot kata (*term*) *t* dari suatu dokumen (*document*) *d* adalah dengan menghitung jumlah kata *t* dalam dokumen *d*, pembobotan ini disebut kemunculan kata (*term frequency*) *TF*. Kelemahan dari *term frequency* adalah semua kata memiliki bobot yang sama pentingnya. Salah satu solusi dari kelemahan ini adalah memberikan bobot yang tinggi untuk kata yang kemunculannya sedikit di banyak dokumen. Hal ini dikarenakan kata yang sedikit muncul di banyak dokumen dianggap penting. Untuk memberatkan bobot kata yang kemunculannya sedikit di banyak dokumen pada umumnya menggunakan *inversed document frequency* (*IDF*). Penggabungan bobot *TF* dan *IDF* dilakukan dengan cara perkalian, penggabungan dilakukan agar didapatkan bobot campuran suatu *term* dari setiap dokumen [4]. Persamaan *IDF* dapat dilihat pada (1) dan Persamaan *TF-IDF* dapat dilihat pada (2). Persamaan (1) dan (2) dikutip dari [4]. Berikut adalah persamaan *IDF*:

$$idf = \log \frac{n}{df_t} \quad (1)$$

dimana:

- n adalah jumlah dokumen yang diamati
- df_t adalah jumlah dokumen yang mengandung *term* t.

$$w_{td} = tf_{td} . \log \frac{n}{df_t} \quad (2)$$

dimana:

- w_{td} adalah bobot *term* t
- tf_{td} adalah frekuensi *term* t pada dokumen d
- $\log(n/df_t)$ adalah *IDF*
- n adalah jumlah dokumen yang diamati
- df_t adalah jumlah dokumen yang mengandung *term* t.

C. Vector Space Model Algorithm

Vector Space Model adalah suatu model yang digunakan untuk memodelkan suatu dokumen. Untuk mengimplementasikan metode-metode klasifikasi dokumen teks, diperlukan suatu transformasi yang dapat mengubah teks-teks *digital* menjadi suatu model yang lebih efisien dan dimengerti sehingga proses analisa dapat dilakukan [7]. *Vector space model* adalah salah satu pendekatan yang paling banyak digunakan dalam merepresentasikan dokumen teks [8]. Representasi kumpulan dokumen sebagai vektor disebut *vector space model*, sebagai contoh V_d adalah vektor dari dokumen *d*. Vektor tersebut memiliki fitur berupa nilai atau bobot dari *term* pada dokumen tersebut [4]. Untuk menghindari vektor dengan fitur yang besar namun tidak penting, kata yang dijadikan fitur hanya apabila muncul pada data *training* minimal sebanyak tiga kali, atau apabila kata tersebut bukan *stopword* [9]. Fitur dari *vector space model* dapat dilihat pada (3). Berikut adalah persamaan dari *vector*

space model:

$$V_d = \{t_1, t_2, ...t_n\} \quad (3)$$

dimana:

- V_d adalah vektor dari dokumen d
- t_i adalah nilai *term* ke-i dari dokumen d.

D. Manhattan Distance Algorithm

Manhattan distance adalah metode pengukuran jarak dua vektor. *Manhattan distance* menghitung jumlah dari perbedaan fitur antara dua vektor dengan n dimensi. Nilai yang dijumlahkan adalah nilai absolut dari masing-masing fitur yang dihitung perbedaannya [10]. Berikut adalah persamaan *Manhattan distance*:

$$ManhDis(p,q) = \| p - q \| = \sum_{i=1}^{n} | p - q | \quad (4)$$

dimana *p* dan *q* adalah vektor dengan persamaan:

$$p = (p_1, p_2 .... p_n)$$
$$q = (q_1, q_2 .... q_n) \quad (5)$$

E. Manhattan Similarity Algorithm

Banyak nilai jarak yang merupakan nilai absolut, tapi apabila ingin dihitung kemiripannya, tentu kemiripannya yang dilihat. Sebagai contoh, apabila ada dua *string* dengan ukuran 10^6 dan memiliki perbedaan hanya 1000 *bits*, maka kedua *string* tersebut dapat ditetapkan memiliki kemiripan yang lebih tinggi dibanding dengan dua *string* dengan panjang 1000 *bits* namun memiliki perbedaan sebesar 100%. Dari sinilah dilakukan normalisasi jarak menjadi jarak relatif agar dapat dihitung kemiripan berdasarkan jarak [11]. Berikut adalah persamaan *Manhattan similarity*:

$$ManhSim(p,q) = 1 - \frac{ManhDis(p,q)}{n} \quad (6)$$

dimana:

- n adalah ukuran dari vektor atau jumlah *term*
- p adalah vektor
- q adalah vector.

Manhattan similarity adalah metode pengukuran kemiripan antara dua vektor dengan menggunakan nilai *Manhattan distance*. Pembagian nilai *Manhattan distance* dengan *N* adalah untuk menormalisasikan jarak menjadi jarak relatif. Pengurangan angka satu dengan jarak relatif bertujuan agar nilai yang didapat memiliki nilai maksimal satu.

F. K-Means Clustering Algorithm

Tujuan dari algoritma *K-means* adalah untuk membagi *M* titik dengan *N* dimensi ke dalam *k cluster* agar menghasilkan nilai *sum of squares* yang minimal. Untuk mendapatkan hasil yang memiliki nilai *sum of squares* yang minimal pada semua partisi hampir tidak realistis, kecuali ketika *M* dan *N* berukuran kecil dan *k*=2. Oleh karena itu, yang dicari adalah solusi "*local*" *optima* yang dicapai dengan tidak ada perpindahan titik ke *cluster* lain yang menyebabkan pengurangan nilai *sum of squares* [12].

Perhitungan rata-rata jarak antara data-data yang ada terhadap *centroid* menggunakan *Mean Squared Error (MSE)* yang dikutip dari [13] dapat dilihat sebagai berikut:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \| v_c - v_i \|^2 \quad (7)$$

dimana:

- n adalah jumlah data pada suatu *cluster*
- v_c adalah vektor *centroid*
- v_i adalah vektor data ke-i.

Langkah-langkah yang digunakan dalam pembuatan cluster menggunakan *K-means* adalah:

1. Tentukan jumlah *cluster* (*k*).
2. Tentukan secara acak data sejumlah *k* sebagai titik awal *centroid*.
3. Hitung rata-rata jarak antara data-data yang ada terhadap *centroid*, alokasikan data-data ke dalam *cluster* yang memiliki jarak terdekat dengan *centroid*-nya.
4. Kembali ke langkah 3 apabila masih ada data yang berpindah *cluster* atau terjadi perubahan nilai *cluster* atau iterasi sudah melebihi nilai maksimal iterasi yang ditentukan.

III. ANALISIS DAN PERANCANGAN

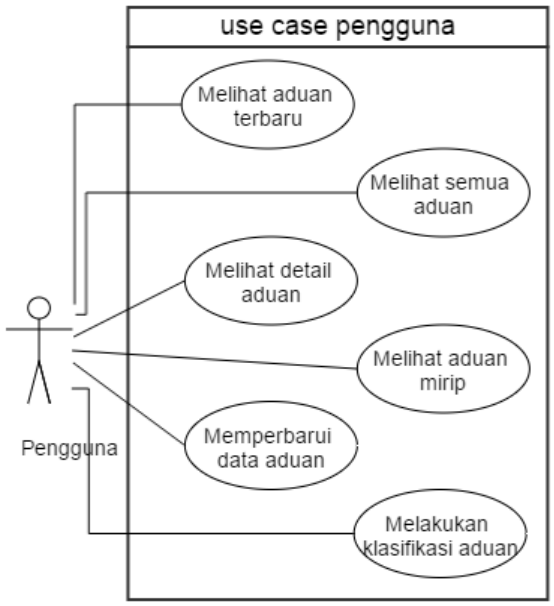
A. Deskripsi Umum Sistem

Pada Tugas Akhir ini akan dibangun modul aplikasi perangkat bergerak Suara Warga (SURGA) Kota Kediri yang memiliki kemampuan untuk mengklasifikasi aduan serta mendeteksi kemiripan suatu aduan dengan aduan lainnya. Aplikasi ini juga dapat menampilkan aduan dengan memasukkan kata kunci, klasifikasi, atau tanggal aduan yang ingin dicari. Aplikasi ini merupakan modul dari aplikasi utama "Operator Surga". Definisi modul yang digunakan adalah aplikasi hanya dapat dibuka oleh aplikasi utama dan tidak dapat dibuka secara langsung dari *launcher android*. Terdapat *class-interface* dari aplikasi utama yang diimplementasikan oleh modul ini agar dapat menjadi modul dari aplikasi utama.

B. Kasus Penggunaan Sistem

Kasus Penggunaan sistem merupakan diagram kebutuhan yang menggambarkan fungsionalitas sistem dan aktor-aktornya. Pengguna dapat melihat aduan terbaru, melihat semua aduan, melihat detail aduan, melihat aduan mirip, memperbarui data aduan dan melakukan klasifikasi aduan. Melihat aduan terbaru merupakan kasus penggunaan untuk menampilkan 15 aduan terbaru pada aplikasi. Melihat semua aduan merupakan kasus penggunaan untuk menampilkan semua aduan, Melihat detail aduan merupakan kasus penggunaan untuk menampilkan detail dari suatu aduan. Melihat aduan mirip merupakan kasus penggunaan untuk menampilkan aduan yang mirip dengan aduan yang dipilih. Memperbarui data aduan merupakan kasus penggunaan untuk mengambil aduan baru dari *server*. Melakukan klasifikasi

aduan merupakan kasus penggunaan untuk membuat klasifikasi aduan di *database* lokal. Kasus Penggunaan sistem dapat dilihat pada Gambar 1.

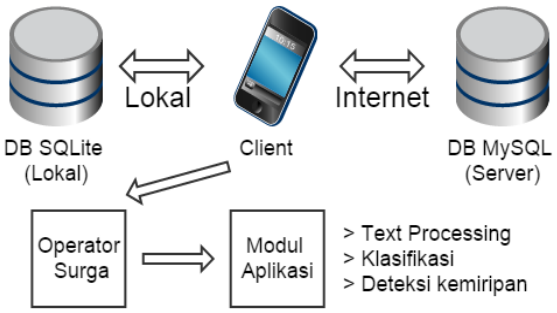


Gambar 1 Diagram Kasus Penggunaan

C. *Perancangan Arsitektur Umum Sistem*

Penjelasan arsitektur sistem yang terlihat pada Gambar 2 adalah sebagai berikut:

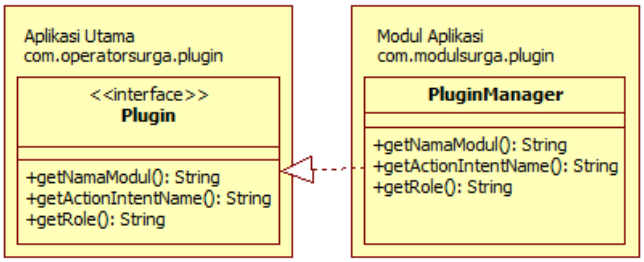
- *Server* adalah *database* yang menyimpan semua aduan pada SURGA Kota Kediri.
- *Client* dapat mengambil data aduan dari *database server* yang kemudian disimpan di *database* lokal.
- Modul aplikasi adalah modul dari aplikasi utama "Operator Surga" yang telah terpasang pada *client*.
- Modul aplikasi menggunakan data aduan dari *database* lokal untuk melakukan klasifikasi dan deteksi kemiripan aduan.



Gambar 2 Arsitektur Umum Sistem

D. *Perancangan Modularitas*

Perancangan modularitas dilakukan dengan mengimplementasikan *interface* milik aplikasi utama. Detail dari *interface* yang diimplementasikan dapat dilihat pada Gambar 3.



Gambar 3 Diagram Kelas Modularitas.

IV. IMPLEMENTASI

Implementasi dilakukan dengan pembuatan database yang terdiri dari tabel berisi aduan dari server, serta tabel berisi nilai-nilai hasil perhitungan klasifikasi dan deteksi kemiripan. Untuk pengambilan data pada *database* lokal, dilakukan implementasi *query* yang berfungsi mengambil data dari *database* lokal sesuai dengan kebutuhan modul aplikasi. Pengambilan data diimplementasikan dengan memodelkan bentuk aduan menjadi *JavaScript Object Notation* (JSON) yang kemudian di-*parse* setelah diambil *client*. Data aduan yang berhasil diambil diproses nilai *TF-IDF* untuk setiap *term*-nya dan disimpan ke dalam *database*.

Clustering atau klasifikasi dilakukan dengan menggunakan nilai dari aduan yang telah diproses dan disimpan sebelumnya. Klasifikasi dilakukan dengan mengambil sejumlah aduan yang dijadikan *centroid* yang kemudian dihitung rata-rata jarak masing-masing *centroid* dengan aduan-aduan lainnya. Perubahan *centroid* dilakukan apabila rata-rata jarak yang dilakukan lebih kecil dari *centroid* sebelumnya. Klasifikasi selesai apabila tidak ada aduan yang berpindah *cluster* atau batas iterasi maksimal telah dicapai. Setelah klasifikasi selesai, maka dicari *term* dengan bobot tertinggi pada masing-masing *cluster* yang berfungsi sebagai label dari klasifikasi tersebut.

Pendeteksian aduan mirip dilakukan setelah klasifikasi terhadap semua aduan telah selesai. Dari setiap *cluster* diambil satu aduan yang terdekat dengan *centroid* dan aduan tersebut dijadikan *medoid*. Setelah *medoid* didapatkan, aduan yang terdapat pada suatu *cluster* dibandingkan dengan *medoid* dari *cluster* tersebut. Perbandingan dilakukan dengan menghitung jarak antara kedua aduan menggunakan *Manhattan distance* dan dicari tingkat kemiripannya dengan *Manhattan similarity*. Pembuatan antarmuka aplikasi dibuat dalam kode XML, halaman utama modul aplikasi dapat dilihat pada Gambar 4.

Modularitas diimplementasikan dengan cara meng-*import library* dari kelas utama. Selain itu, modul aplikasi juga menambahkan *file MANIFEST.MF* yang menyediakan informasi kepada kelas utama informasi kelas yang digunakan oleh modul aplikasi dalam mengimplementasikan *interface*. Modul aplikasi juga mengimplementasikan kelas *interface* dari aplikasi utama. Setelah itu meletakkan *file* JAR yang dibutuhkan ke dalam *folder Plugin* pada media penyimpanan perangkat lunak bergerak.

Gambar 4 Halaman Utama Aplikasi

V. PENGUJIAN DAN EVALUASI

Pengujian fungsionalitas dilakukan dengan menyiapkan sejumlah skenario penggunaan modul aplikasi. Hasil pengujian fungsionalitas menunjukkan bahwa semua fungsionalitas modul aplikasi berhasil. Pengujian klasifikasi dilakukan setelah modul aplikasi dapat mengambil data aduan dari *server*. Setelah klasifikasi dibentuk, dilakukan deteksi kemiripan antara *medoid* dari suatu *cluster* dengan aduan di dalamnya, proses ini menghasilkan nilai *Manhattan similarity* antara aduan dengan *medoid* dari *cluster* aduan tersebut. Setelah deteksi kemiripan selesai, dilakukan observasi secara manual untuk menentukan nilai minimal dari *Manhattan similarity* antar aduan yang digunakan sebagai acuan dari mirip-tidaknya suatu aduan. Penentuan nilai *k* dan jumlah iterasi yang akan digunakan dilakukan dengan cara mencari nilai *MSE* terkecil dari semua skenario yang tersedia. Nilai *k* yang terpilih adalah *k*=4 dan iterasi maksimal 500. Setelah dilakukan klasifikasi dan deteksi kemiripan, dilakukan observasi secara manual untuk menentukan nilai minimal untuk menentukan tingkat kemiripan dari dua aduan yang dibandingkan. Tingkat kemiripan minimal yang digunakan untuk menentukan mirip tidaknya kedua aduan adalah 0,9993. Tingkat kemiripan minimal adalah nol dan maksimal adalah satu. Pengujian akurasi untuk tingkat kemiripan minimal 0,9993 dilakukan dengan metode kuisioner. Kuisioner berisi 20 aduan yang ditetapkan mirip dan 20 aduan yang ditetapkan tidak mirip oleh aplikasi, responden menyatakan aduan yang ditetapkan oleh aplikasi benar atau tidak. Hasil pengujian akurasi menunjukkan bahwa akurasi untuk penetapan aduan tidak mirip adalah 100% dan penetapan aduan mirip adalah 90%. Hasil pengujian fungsionalitas dapat dilihat pada Tabel 1.

Tabel 1 Hasil Pengujian

Uji coba fungsi	Hasil pengujian
Melihat aduan terbaru	Berhasil
Melihat semua aduan	Berhasil
Melihat detail aduan	Berhasil
Melihat aduan mirip	Berhasil
Memperbarui aduan baru	Berhasil
Melakukan klasifikasi aduan	Berhasil
Melakukan deteksi kemiripan aduan	Berhasil
Menampilkan aduan yang telah disaring	Berhasil

VI. KESIMPULAN

Berdasarkan hasil uji coba yang telah dilakukan, terdapat beberapa kesimpulan yang bisa diambil, yaitu:

1) Fungsionalitas melihat aduan, mencari aduan dan menampilkan aduan yang mirip dapat dijalankan dengan baik.
2) Klasifikasi aduan dilakukan dengan metode *K-means clustering* dengan nilai *k*=4 dan iterasi maksimal 500. Kemudian dilakukan penghitungan jarak antar aduan dengan *medoid* dari masing-masing *cluster*, aduan akan dimasukkan ke dalam *cluster* yang terdekat jaraknya dengan *medoid cluster* tersebut.
3) Pendeteksian kemiripan dilakukan dengan cara menghitung jarak antar kedua aduan yang dimodelkan menjadi vektor dan dihitung menggunakan *Manhattan similarity*. Jarak kedua vektor aduan merupakan tingkat kemiripan. Tingkat kemiripan minimal adalah nol dan maksimal adalah satu. Apabila tingkat kemiripan kedua aduan lebih besar dari 0,9993, maka aduan dianggap mirip.
4) Aduan yang telah disaring ditampilkan dengan melakukan proses *query* pengecekan aduan untuk mengecek apabila suatu aduan memiliki nilai jarak lebih dari 0,9993 maka aduan tersebut tidak ditampilkan karena dianggap mirip.
5) Modularitas diimplementasikan dengan cara meng-*import library* dari kelas utama. Modul aplikasi juga mengimplementasikan kelas *interface* dari aplikasi utama. Setelah itu meletakkan *file* JAR yang dibutuhkan ke dalam *folder Plugin* pada media penyimpanan perangkat lunak bergerak.

DAFTAR PUSTAKA

[1] Badan Pusat Statistik Kota Kediri. [Online]. http://kedirikota.bps.go.id/LinkTabelStatis/view/id/4 , diakses pada 29 Juli 2015.

[2] Andani Achmad, Amil Ahmad Ilham, dan Herman , "Implementasi Algoritma Term Frequency - Inverse Document Frequency dan Vector Space Model untuk Klasifikasi Dokumen Naskah Dinas," *Prosiding Konferensi Nasional Forum Teknik Elektro Indonesia (FORTEI 2012)*, 2012.

[3] M. I. Azis, Development Program Application To The Measurement Of Document Resemblance Text Mining, TF-IDF, and Vector Space Model Algorithm, Gunadarma University, 2010.

[4] Christopher D. Manning, Prabhakar Raghavan, dan Hinrich Schütze, *Introduction to Information Retrieval*. California: Stanford University, 2008.

[5] Megan Chenoweth dan Min Song, "Text Categorization dalam Encyclopedia of Data Warehouse & Data Mining," *IGI Global*, pp. 1936-1941, 2009.

[6] Fadillah Z. Tala, "A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia", Universiteit van Amsterdam, The Netherlands, 2003.

[7] A. Z. Arifin, Roby Darwanto, Dini Adni Navastara, dan Henning Titi Ciptaningtyas, "Klasifikasi Online Dokumen Berita Dengan Menggunakan Algoritma Suffix Tree Clustering," *Seminar Sistem Informasi Indonesia (SESINDO2008)*, Desember 2008.

[8] Aini Rachmania Kusumaagama Fuddoly, Klasifikasi Kategori dan Identifikasi Topik pada Artikel Berita Berbahasa Indonesia, Institut Teknologi Sepuluh Nopember, Surabaya, 2011.

[9] T. Joachims, C. Nedellec, dan C. Rouveirol, "Text categorization with support vector machines: learning with many relevant," *10th European Conference on Machine Learning*, 1998.

[10] Abul Hasnat, Santanu Halder, D. Bhattacharjee, M. Nasipuri, dan D. K. Basu, "Cooparative Study of Distance Metrics For Finding Skin Color Similarity of Two Color Facial Images," *National Conference on Advancement of Computing in Engineering Research (ACER 13)*, 2013.

[11] Ming Li, Xin Chen, Xin Li, Bin Ma, dan Paul M. B. Vitányi, "The Similarity Metric," *IEEE Transactions On Information Theory*, vol. 50, no. 12, pp. 3250-3264, 2004.

[12] J. A. Hartigan dan M. A. Wong, "A K-Means Clustering Algorithm," *Journal of the Royal Stastical Society. Series C*, vol. 28, no. 1, pp. 100-108, 1979.

[13] S. Makridakis dan M. Hibon, "Evaluating Accuracy (Or Error) Measures," INSEAD, Fontainebleau, 1995.